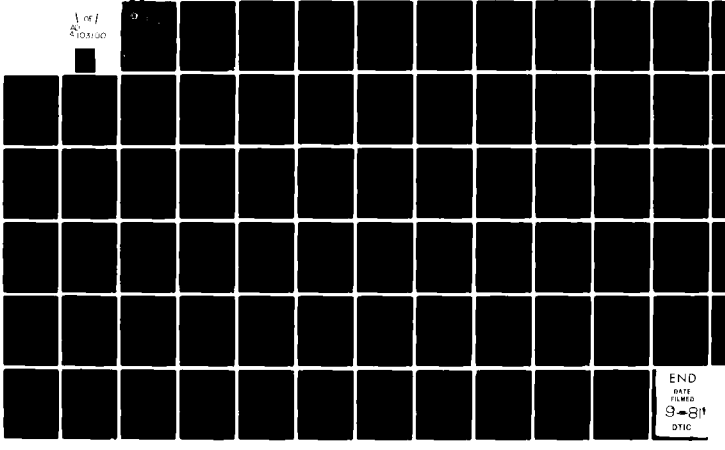
AD-A103 100 KANSAS STATE UNIV MANHATTAN DEPT OF COMPUTER SCIENCE F/G 9/2
RESEARCH IN FUNCTIONALLY DISTRIBUTED COMPUTER SYSTEMS DEVELOPME--ETC(U)
UNCLASSIFIED MAY 76 F J MARYANSKI, V E WALLENTINE DAAG29-76-G-0108
CS-76-13 NL



END
DATE FILMED
9-81
DTIC

AIRMICS

Army Institute for Research in Management Information and Computer Science

313 Calculator Bldg.
GA Institute of Technology
Atlanta, GA 30332

Technical Report

RESEARCH IN FUNCTIONALLY DISTRIBUTED COMPUTER SYSTEMS DEVELOPMENT

Kansas State University

Virgil Wallentine

Principal Investigator

Approved for public release; distribution unlimited

VOLUME XVIII

LANGUAGE SPECIFICATION FOR A DISTRIBUTED DATA MANAGEMENT SYSTEM

U.S. ARMY COMPUTER SYSTEMS COMMAND FT BELVOIR, VA 22060

(9) Interim rept.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: (6) Research in Functional Distributed Computer Systems Development. Volume XVIII.	2. GOVT ACCESSION NO.: AD-A103100
4. TITLE: LANGUAGE SPECIFICATION FOR A DISTRIBUTED DATA BASE MANAGEMENT SYSTEM. A103099	3. RECIPIENT'S CATALOG NUMBER
	5. TYPE OF REPORT & PERIOD COVERED: Interim
	6. PERFORMING ORG. REPORT NUMBER: (14) CS-76-13
7. AUTHOR(s): (10) Fred Maryanski (J.)	8. CONTRACT OR GRANT NUMBER(s): (15) DAAG 29-76-G-0108
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Kansas State University, Department of Computer Science, Manhattan, KS 66506	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: US Army Research Office, P O Box 12211, Research Triangle Park, NC 27700 (11)	12. REPORT DATE: May 1976 13. NUMBER OF PAGES: 76 pages
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): US Army Computer Systems Command, Attn: CSCS-AT, Ft. Belvoir, VA 22060 (12) 77	15. SECURITY CLASS. (of this report): Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

DDBMS

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

-over-

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

391123

-ABSTRACT-

This document is a proposal for a distributed data base management system (DDBMS). It represents the first phase of the DDBMS design portion of Grant 108. It is very important to note that this document is a proposal and also that the next phase of the design is the development of the functional specifications of the DDBMS. Therefore, it is essential that all interested parties respond with any corrections, additions, deletions, suggestions, etc. by July 1, 1976.

As it can easily be observed from this report, the implementation of the complete DDBMS will be an enormous task. Estimates range from 7 to 20 up to 50 person years of effort. A natural course is to design a full scale system and proceed with the implementation in an incremental manner. The implementation of a minimal prototype should be achieved as soon as possible for purposes of feasibility studies, testing, and morale. Another important consideration is that based upon the current resource allocation to the data base portion of Grant 108, it is unlikely that the Special Features described in Chapter VIII can be included in the initial DDBMS design.

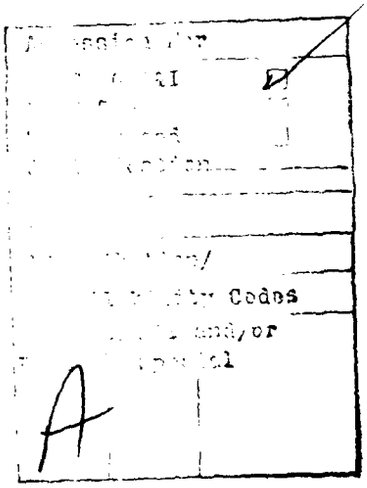
Language Specification for a Distributed Data Base Management System[1]

Technical Report
CS76-13

Fred J. Maryanski

May, 1976

Computer Science Department
Kansas State University
Manhattan, Kansas 66506



This work is supported by the U.S. Army Research Office Grant No. DAAD-20-76-G-0108.

TABLE OF CONTENTS

PREFACE

This document is a proposal for a distributed data base management system (DDBMS). It represents the first phase of the DDBMS design portion of Grant 108. It is very important to note that this document is a proposal and also that the next phase of the design is the development of the functional specifications of the DDBMS. Therefore, it is essential that all interested parties respond with any corrections, additions, deletions, suggestion, etc. by July 1, 1976.

As it can easily be observed from this report, the implementation of the complete DDBMS will be an enormous task. Estimates range from 7 to 20 up to 50 person years of effort. A natural course is to design a full scale system and proceed with the implementation in an incremental manner. The implementation of a minimal prototype should be achieved as soon as possible for purposes of feasibility studies, testing, and morale. Another important consideration is that based upon the current resource allocation to the data base portion of Grant 108, it is unlikely that the Special Features described in Chapter VIII can be included in the initial DDBMS design.

Again, any inputs regarding the specification design or implementation will be greatly appreciated.

ABSTRACT

This report describes the features of a proposed distributed data base management system (DDBMS). DDBMS is based upon the CODASYL specifications. It is intended for use in a multi-computer environment in which many of the machines will be mini-computers produced by different vendors. This document concentrates on the user level view of the system. That is, the features of the DDBMS are presented, rather than the functional specification. The main areas of emphasis are the schema data definition language, sub-schema data definition language, data manipulation language, device media control language, and utility routines. Some further enhancements of DDBMS are suggested and a comparison is made with existing CODASYL-based DBMS systems.

Introduction

In this report, a set of specifications are presented for a distributed data base management system (DDBMS). The DDBMS is based upon the CODASYL specifications [1]. The DDBMS is targeted for a network environment. The machines comprising the network may be of various types and sizes. In DDBMS, a user may initiate a program from any computer in the network, have it executed on another (or the same processor) and access data on storage devices at any node in the network (provided security restrictions are satisfied). A discussion of the characteristics of distributed data base systems is given in Reference [2]. The communication system for the network is described in Reference [3].

This document presents the application programmer view of the DDBMS. The physical structure of the system is transparent to the user. The application programmer need only to concern herself/himself with the language features as listed in this report. The languages of the DDBMS are identical in both network and single machine environments. Implementation considerations are treated in Reference [4].

The essential concepts of data base management systems are initially discussed to form a basis for the language definitions. The DDBMS contains four languages: a schema data definition language to describe the structure of an entire data base: a subschema data definition language to select the portion of the data base visible to an application program; a data manipulation language to operate on a data base; and a device media control language to map the logical data base onto physical storage. The syntax of each of the four languages is specified in this report along with their functional characteristics.

In order to facilitate the use of DDBMS by a wide band of users ranging from clerical personnel to the data base administrator, a substantial number of utility programs have been included in the system. The capabilities of the

DDBMS utilities are listed in Chapter VII of this report.

Chapter VIII presents a list of special features to be added to future versions of DDBMS. These enhancements include query and data base administrator languages, a debugging facility, a report generator, a data dictionary facility, and an interactive data base design language. An additional topic treated in this report is the compatibility of DDBMS and relational data models. Features have been included in DDBMS with the intention of constructing a means of bridging the differences between DDBMS and relational data bases at some later time.

The final chapter of the report concentrates on the differences between DDBMS the CODASYL specifications [1], and four existing CODASYL based DBMS packages, DEC's DBMS-10, UNIVAC's DMS-1100, Honeywell's (Xerox) EDMS, and Cullinane's IDMS. The comparison shows that DDBMS is the closest system to the original CODASYL specifications.

II. DBMS Concepts

A. Definitions

1. Schema - complete description of a data base.
2. Sub-Schema - description of data base known to a particular program.
3. Schema DDL - Language for describing a data base.
4. Sub-Schema DDL - Language for describing that part of a data base known to a program.
5. DML - Language used by the programmer to cause data to be transferred by the program to the data base.
6. Data-Item - smallest unit of named data.
7. Data-Aggregate - collection of data-items; either vector or repeating group.
8. Record - collection of data-items or data-aggregates. There may be an arbitrary number of occurrences of a given record type in a data base.

9. Set - collection of record types.
10. Realm - sub-division of addressable secondary storage space.
11. Data base - record occurrences, set occurrences, and realms controlled by a specific schema.
12. Data Base Key - unique name assigned to each record occurrence. Used to map to the physical location of the record.

B. Implementation Considerations

The DDL's and the DML are all COBOL based languages. The DDL's are distinct languages which strongly resemble COBOL Data Division statements. Separate compilers are required for the schema and sub-schema DDL. These compilers produce object versions of the schema and sub-schema which are essentially templates which define the layout and logical organization of the data base.

The DML is actually an extension of COBOL to facilitate accessing of the data base. A pre-processor can be used to map the DML statements into standard COBOL for processing by the COBOL compiler.

The organization of a CODASYL DBMS is depicted in Figure 1. The actions resulting from a user data manipulation language (DML) command are discussed below as a means of illustrating the workings of the system.

1. A DML command is encountered in the application program. A call to the DBMS is then issued.
2. The DBMS analyzes the call and verifies the request against the object versions of the schema and sub-schema.
3. The contents of system buffers are checked.
4. If necessary, the DBMS requests that the operating system perform a physical I/O transfer.
5. The operating system controls the I/O operation of 6.
6. Data is transferred between secondary storage and system buffers by the operating system.

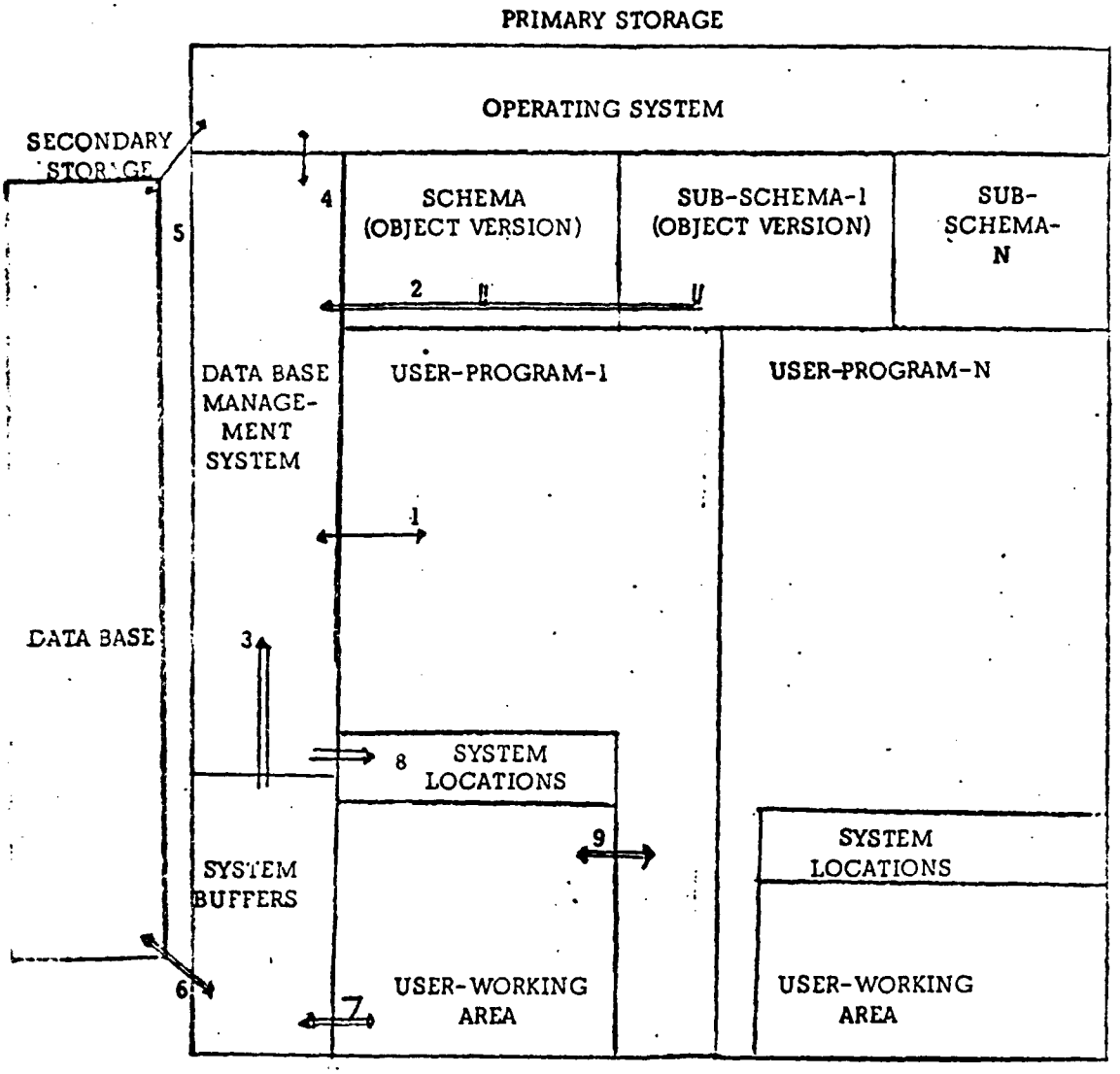


Figure 1
DBMS Memory Layout
and Operation

→ Control flow

⇒ Data flow

7. The DBMS transfers data between system buffers and the User Working Area (UWA) as required.

8. The DBMS provides status information on the recently completed operation.

9. The data in the UWA may be operated upon in any manner by the application program.

The operation of a DBMS distributed over several machines is not described explicitly in this report since the report is concerned with the user level specifications of the DBMS. At the user level the means of implementation of the DBMS in terms of the machine configuration is transparent. For a discussion of the basic features of distributed data bases see Reference [2].

C. Differences Between Schema and Sub-Schema

As previously stated a schema is a description of a universal data base, while a sub-schema is the user's view of the data base. Since the sub-schemata available to the individual user are determined by the data base administrator (whose function is discussed in Section II.E), the concept of a sub-schema entails intrinsic security mechanisms.

In DDBMS, a sub-schema is for the most part a subset of the schema. Realms, sets, and records may be copied in toto from the schema. At any level, lower level entities may be omitted. For example, sets may be omitted from realms, records from sets, etc. The only other differences between a sub-schema and its corresponding schema are that data aggregates may be formed or redefined at the record level provided that the data items are not reordered, and that names may be changed.

D. User Working Area

Each application program has a unique UWA which serves as a common data area shared by the program and the DBMS. The UWA is used in the transfer of data between the program and the data base. The form of the data in the UWA is derived from the sub-schema. During execution, the UWA holds the current

(most recently accessed) data for all set and record types defined by the sub-schema, as well as necessary status information.

E. The Data Base Administrator

In any system as complex and important as a DBMS, there must be a clearly defined position of responsibility to oversee its operation. The Data Base Administrator (DBA) is responsible for the organization, monitoring, and reorganization of the data base. The DBA defines the schema of the data base and controls access to it through the definition of sub-schemata for individual applications. Many of the utility functions mentioned in the Chapter VII are intended for usage of the DBA. A list of the principal DBA functions is provided below.

1. Model the data base using the Schema DDL.
2. Assign privacy locks and keys.
3. Assign realms to devices by specifying the DMCL (see Chapter VI).
4. Load the data base.
5. Specify the sub-schema DDL for an application.
6. Assign privacy keys to application programs.
7. Monitor data base performance and security.
8. Modify the schema.
9. Supervise data migration and archiving.
10. Restructure the data base.
11. Perform garbage collection.
12. Reassign realms.
13. Tune the data base by reorganizing the placement of data in order to increase system performance.

F. Privacy

It is proposed that DDBMS implement an extensive range of privacy features. The basic security mechanism is through privacy locks which are specified in

the schema and sub-schema, and privacy keys which must be provided by the application program to access data which have privacy locks assigned. Privacy locks can be assigned at all levels of the schema and sub-schema.

Locks can be assigned for the following specific functions.

1. At the schema or sub-schema level, locks may be specified against the alteration or display of the schema or sub-schema itself, against the display of the locks, or against the use of the sub-schema.
2. At the realm level, locks may be specified against the use of the realm for retrieval, update, or any support function.
3. At the record level, locks may be provided against the use of each DML record, data-item, or data-aggregate command.
4. At the set level, locks may be specified against each DML set operation.

A privacy lock is a single value against which the application program-generated privacy key is matched. If the lock and key match, operation continues. Otherwise, a DBA-supplied routine is called to determine the action to be taken. Depending upon the potential severity of the intrusion, this routine may pursue a course of action ranging from merely logging the unsuccessful match to suspending execution of the application program and sending a message to the system console.

G. Data Integrity

Since sub-schemata may overlap it is possible that different applications programs may interact with the same data concurrently. This situation can arise when the DBMS is executing under any operating system that allows multi-tasking. Facilities exist within the DML to allow an application program to be notified when concurrent updating of a record occurrence is taking place.

The USAGE MODE feature allows a program to establish exclusive control over a realm. When data is shared, it is possible for two or more programs

to enter a deadlock state when simultaneously attempting to access shared data items. In DDBMS, a deadlock prevention algorithm is employed to insure proper data access at the record level. Basically, deadlock is avoided by allowing the system to give exclusive control over a set of records to a program that accesses shared data items. This exclusive control of records is transparent to the user and beyond program control. In order to properly maintain lists of shared records, some amount of intertask communication is necessary. However, this communications overhead is substantially less than that accrued if a deadlock detection scheme were used. A deadlock detection algorithm involves rolling back tasks to some deadlock-free state. Rollback of communicating tasks in a network environment has the potential for a substantial amount of communications overhead and also may effect tasks not directly involved in the deadlock situation. For more information concerning the deadlock problem in the DDBMS see Reference [5].

H. Records

1. Data types - DDBMS supports the following data types.

   a. Arithmetic - decimal or binary base, fixed or floating-point scale, real or complex mode.

   b. String - character or bit string.

   c. Data base keys.

   d. Vectors - one dimensional, ordered collection of data items.

   e. Repeating groups - collection of data that appears an arbitrary number of times.

2. Records in Sub-Schemata

The content of a record in a sub-schema must be a subset of the corresponding record in the schema. Records are defined using COBOL Data Division-type statements. Level numbers are used to show the structure within the records. Arrays may have up to three dimensions.

Each data item in a sub-schema record must correspond to a data item or aggregate of that record in a schema or be a new group whose subordinate items correspond to data described for the record in the schema. Rules for correspondence between data-items and data aggregates in schema and sub-schema records are given below.

a. Data items in the schema may only be declared as elementary data items in the sub-schema.

b. A vector of fixed dimension can be mapped into arrays of one, two, or three dimensions.

c. A vector of variable dimensions can be mapped into an array, of one dimension.

d. A repeating group can be mapped into a repeating group.

e. Vectors and repeating groups may be grouped and mapped into new repeating groups in the sub-schema.

f. Data items and aggregates which are components of repeating groups in the schema may only be declared in the sub-schema as components of repeating groups.

3. Record Selection

Records are selected from a database by means record selection expressions. There are three types of record selection expressions.

a. Record selection based on identifiers

Each record in the data base can be retrieved using its database key. If the location mode is CALC (location modes are described in the next section), any record of that type may be retrieved by specifying record name and the value of the data item specified as the CALC key.

b. Record selection based on currency

Throughout the execution of an application program, the

DDBMS maintains the following currency status information on the last record occurrence accessed by the program:

a. each record type (i.e., current of record name)

b. each set type (owner or member) (i.e., current of set-name)

c. each realm (i.e., current of realm name)

d. Last record type accessed (i.e., current of run-unit).

Record selection based on currency allows records whose identifiers are currently saved by the DDBMS to be retrieved if any additional information is required.

c. Positional record selection

Records may be selected based upon their relative position to the current record of set-name. The NEXT, PRIOR, OWNER, FIRST, or LAST member of a set can be retrieved in this manner.

4. Location Mode

Three access methods can be employed to locate data.

a. Direct - retrievel using the data base key.

b. CALC - key transformation (hash coding) used to map a specified data item into the data base key.

c. VIA set name - retrievel is based upon set relationships. In order to perform this operation, first the proper set occurrence must be selected, and then the correct record occurrence must be chosen from the members of the set. In order to select a set occurrence the owner record occurrence must be selected. The correct member record occurrence is then selected by means of an argument provided to discriminate among the member records.

I. Sets

1. Concepts

a. Each set must have one owner record type and one or more

member record types in a schema.

b. Any record type may be the owner of one or more sets.

c. Any record type may be a member of one or more sets.

d. Any record type may be both an owner of one or more sets and a member of one or more sets.

e. A record type may not be both owner and member of the same set.

f. A set occurrence is a collection of one or more logically related record occurrences.

g. Each occurrence of a set includes one occurrence of its owner record.

h. A set occurrence which contains only an occurrence of its owner record is known as an empty set.

2. Set Ordering

A set in the schema may have an order specified. This allows the DBMS to control the logical order of the record occurrences within that set. Logical order is independent of physical placement of member records. A listing of possible set orderings is provided below.

a. Sorted - in ascending or descending sequence based upon specified keys.

b. First - newest member is the immediate successor of owner record.

c. Last - newest member is the immediate predecessor of owner record.

d. Next - newest member inserted after a selected set member.

e. Prior - newest member inserted before a selected set member.

f. Immaterial - no order specified.

3. Set Membership

There are four types of membership that a record may have in in a set.

a. Automatic - membership in the set is established by the DBMS when a record occurrence is stored in the database.

b. Manual - membership must be established by a program by means of an INSERT command.

c. Mandatory - once the membership of a record occurrence in a set is established, the record occurrence cannot be removed from the set. Only if an owner record is deleted are the mandatory members removed.

d. Optional - membership is not necessarily permanent. That is, it can be removed from the set.

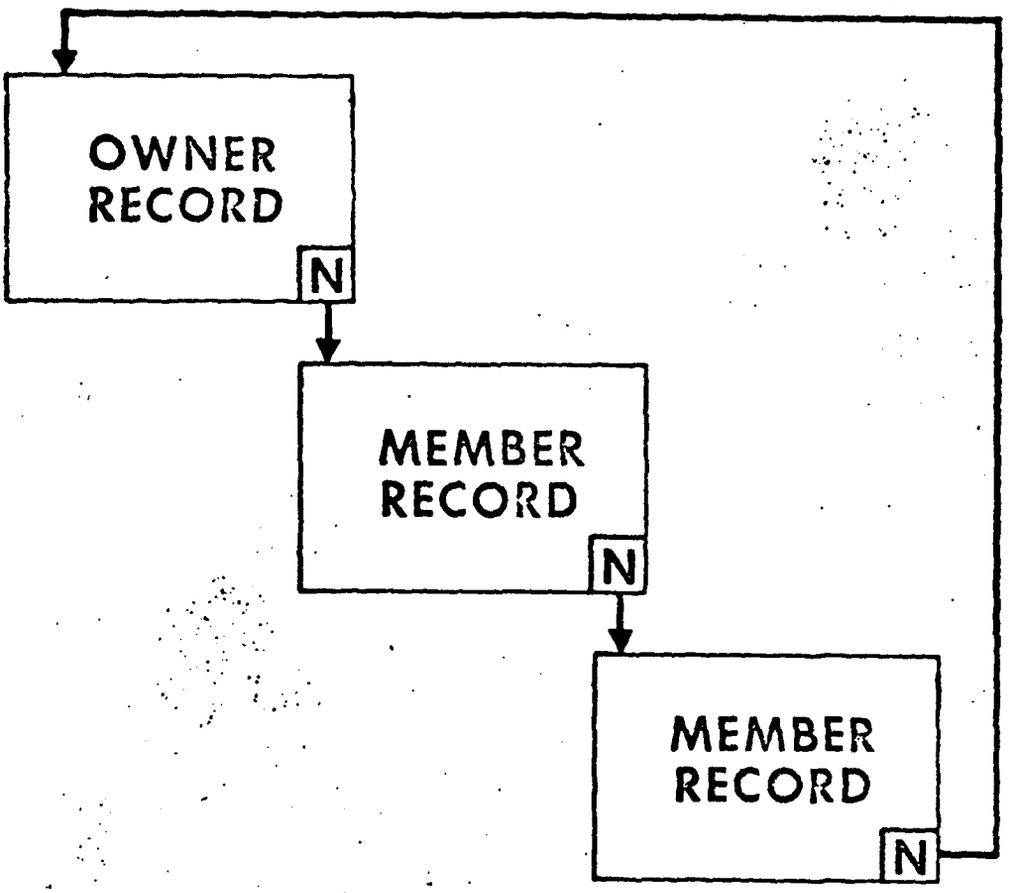
4. Set Mode

Sets may be organized in two basic modes in the proposed DBMS: chains and pointer arrays.

a. Chains - in a chained set the records are linked together using pointers. For each record occurrence in the set there is a pointer to the next (based on set order) record occurrence in the set. Figure 2 illustrates a chained set. Note that the last member points back to the owner. Chains are always linked using Next pointers, but they may also be linked using Prior pointers as shown in Figure 3. In addition, each member record may have a link to the owner record as shown in Figure 4. In all of the above configurations, the pointers to members or the set are database keys.

b. Pointer arrays - pointer arrays are functionally equivalent

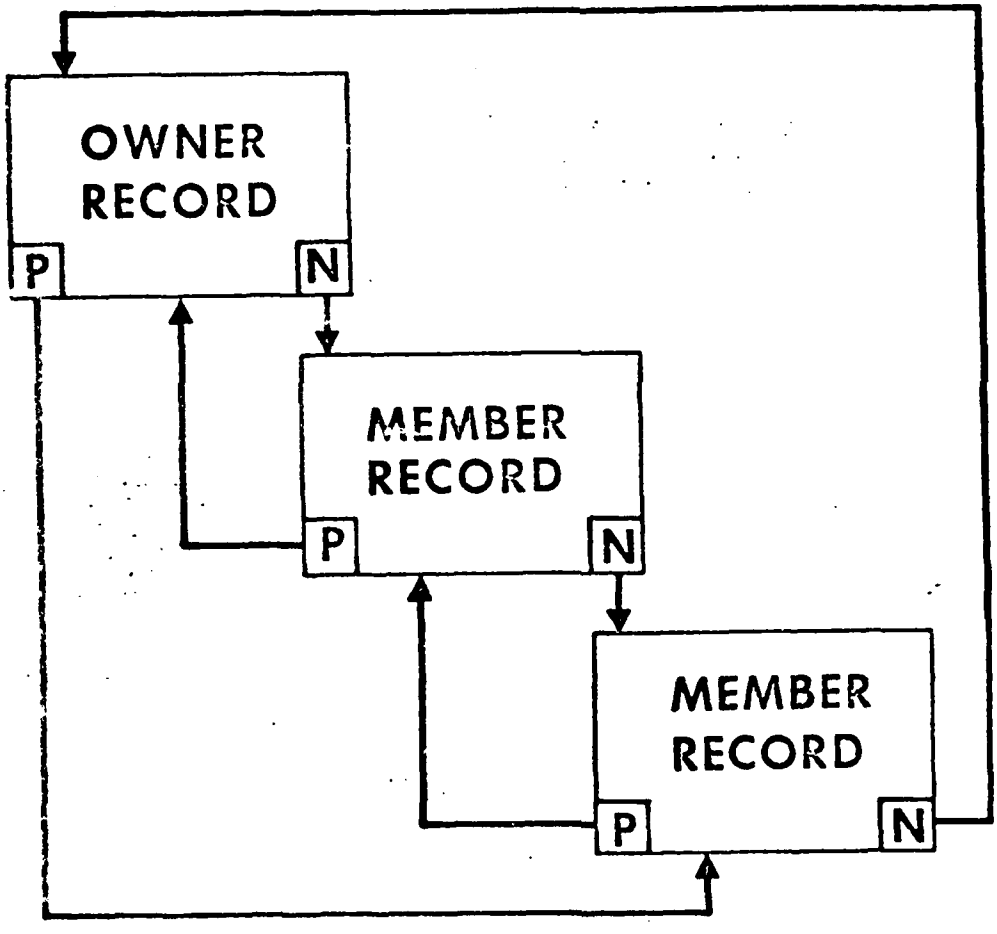
CHAIN WITH NEXT POINTERS



N = NEXT POINTER

FIGURE 2

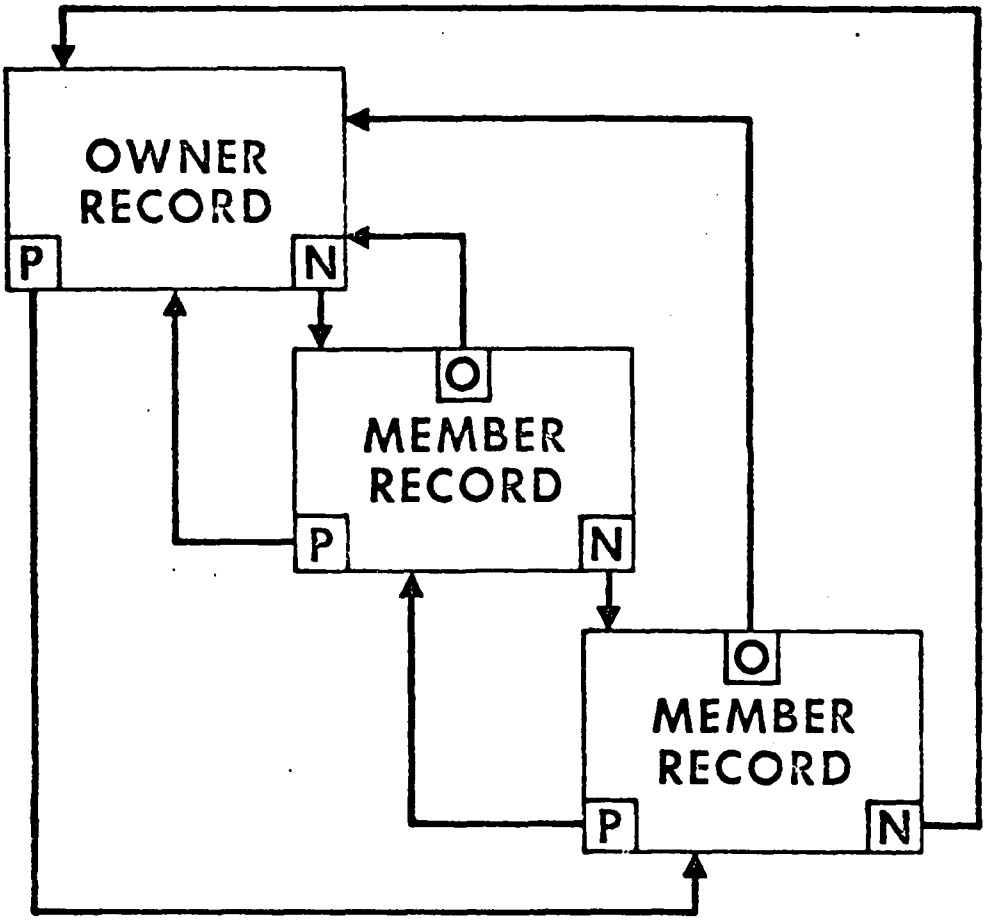
CHAIN
WITH NEXT & PRIOR POINTERS



N = NEXT POINTER
P = PRIOR POINTER

FIGURE 3

CHAIN WITH NEXT, PRIOR & OWNER POINTERS



N = NEXT POINTER
P = PRIOR POINTER
O = OWNER POINTER

FIGURE 4

to chains. In a set declared to have the Pointer Array mode, the Next pointers are collected in a list which is associated with the owner record of the set. Thus the owner record is linked to each member. Each member record contains a pointer back to the owner. Figure 5 illustrates the Pointer Array mode. The list of pointers is maintained in the order declared for the set.

It should be noted that pointer arrays possess all of the capabilities of chains with less overhead. For example, prior pointers are unnecessary with pointer arrays.

III. Schema DDL

The capabilities of the Schema DDL for the DDBMS are outlined in this chapter. The environment of the language is presented along with descriptions of each verb. The verb descriptions indicate the function and the format of the statements.

1. Environment

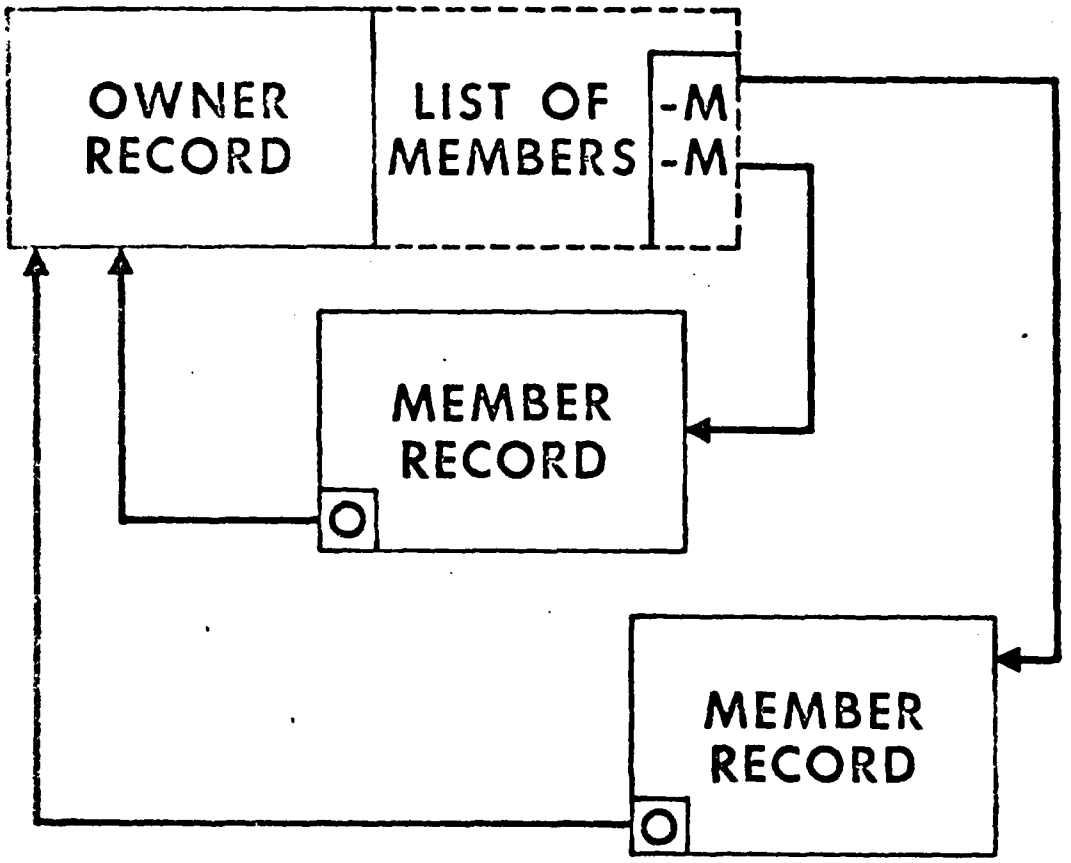
a. Organization

A schema DDL program consists of four distinct sections which must appear in the order stated here. Initially the schema entry must appear to identify the schema. This is followed by realm, set, and record sections, in that order.

b. Lanugage elements

The character set and reserved words for the schema DDL are identical to those given in Reference [1]. The syntax of the language is essentially the same as that of the COBOL Data Division.

POINTER ARRAY



O = OWNER POINTERS

M = MEMBER POINTERS

FIGURE 5

2. Schema Section

The function is to identify the schema of the data base.

The general format is:

SCHEMA IS name

[PRIVACY clause]

where name is an identifier up to 30 characters and is unique among schema names in the data base.

a. PRIVACY clause

Function is to specify privacy locks for the operations modifying, displaying, or changing the privacy locks.

The format is:

PRIVACY LOCK [FOR operations] IS lock

where operations is any combination of ALTER, COPYING or LOCKS.

lock is a literal or procedure name. There may be any number of locks. Separate locks may be specified for each operation.

3. Realm Section

Function is to define a realm and to specify its storage structure.

Format is:

REALM IS name

RANGE IS pn-1 THRU pn-2

[PRIVACY clause]

where name is an identifier;

pn-1 and pn-2 are integer page numbers, such that pn-1 < pn-2 .

a. PRIVACY clause - is identical to that for the schema section except that the operations are EXCLUSIVE RETRIEVAL, EXCLUSIVE UPDATE, PROTECTED RETRIEVAL, PROTECTED UPDATE.

4. Set Section

The function of the set section is to specify the set types available to the sub-schema and application program. Options include specification of the method for distinguishing among sets of the same types and association of privacy locks for the sets.

The set section has two subdivisions: a set description and member description. The member description is optional.

The set description has the following format.

SET clause

MODE clause

ORDER clause

PRIVACY clause

OWNER clause

The Member subentry has the following format:

MEMBER clause

ASC/DES clause

SET SELECTION clause

where

the PRIVACY, ASC/DES, and SET SELECTION clauses are optional.

a. SET clause

The SET clause names the set.

The format is

SET IS name.

where

name is an identifier up to 30 characters in length.

b. MODE clause

Function is to specify the organization of the set.

There are two formats.

i: For chains,

MODE IS CHAIN [LINKED to Prior]

ii: For pointer arrays,

MODE IS POINTER-ARRAY

c. ORDER clause

Function is to specify order of set by indicating insertion point of newest member record.

There are two possible formats.

i: ORDER IS { FIRST / LAST / NEXT / PRIOR / IMMATERIAL }

ii: ORDER IS SORTED [WITHIN RECORD-NAME
BY DATABASE-KEY
DUPLICATES ARE { FIRST / LAST / NOT ALLOWED }]

d. PRIVACY clause

Function is to specify privacy locks for the set.

The format is identical to that for the schema entry, privacy clause except that operations is FIND, OBTAIN, CONNECT, or DISCONNECT.

e. OWNER clause

Function is to name the owner record of the set.

The format is

OWNER IS record-name .

f. MEMBER clause

Function is to name member records of the set.

and specify the type of membership in the set.

The format is

MEMBER name { MANDATORY / OPTIONAL } { AUTOMATIC / MANUAL } [LINKED TO OWNER]

[DUPLICATES ARE NOT ALLOWED FOR identifiers]

where

name is a record;

identifiers is a list of data items in the member record.

g. ASC/DES clause

Function is to specify the sort keys for a sorted set.

The format is

{ ASCENDING / DESCENDING } KEY IS identifiers [DUPLICATES ARE { FIRST / LAST / NOT ALLOWED }]

where

identifiers is a list of data items in the member records

h. SET SELECTION clause

Function is to define the rules for selecting a member of the set for processing.

The format is

SET SELECTION { CURRENT OF SET / LOCATION MODE OF OWNER / [USING (identifiers)] }

where

identifiers is a list of identifiers in the owner record.

5. Record Section

The function of the record section is to indicate the record types available to the sub-schema and application program. The data item format is also specified.

The record section is divided into two portions: the record description and data description.

The format is.

Record Description

RECORD IS name

[LOCATION MODE clause]
WITHIN clause
[PRIVACY clause]

Data Description

[levelnumber]

[PICTURE clause]
[TYPE clause]
[OCCURS clause]
[RANGE clause]
[PRIVACY clause]

where

name is an identifier up to 30 characters in length that is unique among records in the schema;

levelnumber is a two digit positive integer.

a. LOCATION MODE clause

The function of the LOCATION MODE clause is to define a criteria for selecting a record occurrence or placing a record occurrence in a realm.

There are four formats for the LOCATION MODE clause.

1. LOCATION MODE IS DIRECT dbkey

where

dbkey is a database key that must match the database key of the record for a retrieval statement.

ii. LOCATION MODE IS SYSTEM

iii. LOCATION MODE IS CALC USING keys.

DUPLICATES ARE [NOT] ALLOWED

where keys is a list of one or more identifiers used as arguments for the CALC mapping function. If DUPLICATES ARE NOT ALLOWED is specified, no additional occurrences of this record type are allowed to exist if all of its keys are identical to those of an occurrence in the same area.

iv. LOCATION MODE IS VIA setname SET

where setname is a set of which this record is a member.

b. WITHIN clause

Function is to define the realm in which a record occurrence is to be placed.

The format is:

WITHIN realm

where

realm is a realm in the schema;

identifier is any identifier.

c. PRIVACY clause

Function is to specify privacy locks for records,

data items, or data aggregates.

The format is identical to that for schema entry privacy clauses except that:

for records, operations is any DML command;

for data, operations is STORE, GET, MODIFY, or OBTAIN.

d. PICTURE clause

Function is to describe the characteristics of a data item.

Format is:

PICTURE IS character string

where character-string follows standard COBOL rules [1] for picture specification of strings and numbers.

e. TYPE clause

Function is to describe characteristics of a data item.

There are three formats.

i: For a numeric data item

TYPE IS { BINARY / DECIMAL } { FIXED / FLOAT } { REAL / COMPLEX } precision

where

DECIMAL, FIXED, REAL are the default value;

precision is one (FIXED) or two (FLOAT) integers specifying the precision of the number.

ii: For string data,

TYPE IS { BIT / CHARACTER } size

where size is the string length.

iii: For a databasckey

TYPE IS DATABASE-KEY.

f. OCCURS clause

Function is to define a vector or repeating group by specifying the number of times the data item or group occurs within a record.

The format is:

OCCURS count TIMES

where

count is an integer or identifier.

g. RANGE clause

The function of the RANGE clause is to specify upper and lower bounds for the values a data item may assume. The format is

RANGE IS FROM valuel TO value2

where

valuel and value2 are literals of the type corresponding to the data item such that valuel $\leq$ value2.

IV. Subschema DDL

The capabilities of the sub-schema DDL for DDBMS are outlined in this chapter. The organization of the language and the format and function of each verb are presented. The syntax of the sub-schema DDL is given in Appendix B.

1. Organization

A subschema DDL program is divided into four sections which must occur in the sequence specified. The subschema identification section appears first followed by the realm, set, and record sections.

2. Sub-Schema Section

The sub-schema section defines the sub-schema. The general format is

SUB-SCHEMA IS name1 WITHIN name2

PRIVACY LOCK clause

PRIVACY KEY clause

where name1 is an identifier up to 30 characters and is unique among subschema names in the schema;

name2 is a schema name.

a. PRIVACY LOCK clause

Function is to specify privacy locks for the sub-schema.

The format is identical to the PRIVACY clause for schema entry expect that operations is LOCKS, INVOKING, or ALTERING.

b. PRIVACY KEY clause

Function is to specify a key for accessing a schema.

The format is

PRIVACY KEY IS { literal / procedurename }.

3. Alias Section

The alias section provides for the renaming of elements in the sub-schema.

The format is

ALIAS OF name1 [REALM-NAME / SET-NAME / RECORD-NAME] IS name2

where name1 is the a realm, set, or record in the schema;

name2 is a identifier by which the object name1 shall be

identified in the sub-schema;

the phrases REALM-NAME SET-NAME, or RECORD-NAME are necessary if name1 is not unique to the schema.

4. Realm Section

The function of the realm section is to enumerate the realms of the schema to be included in the sub-schema.

The format is

RD { ALL / name }

where

name is a realm in the schema;

ALL means all realms in the schema are included in the sub-schema.

5. Set Section

The purpose of the set section is to enumerate the sets defined in the schema that are to be included in the sub-schema.

The format is

SD { ALL / set1 }

[SET SELECTION clause]

where

set1 is a set in the schema;

ALL means all sets in the schema are included in the sub-schema.

a. SET SELECTION clause

The function is to define the rules governing the

selection of a set occurrence within a sub-schema in order to operate upon a member record.

The format is

SET SELECTION FOR record IS VIA set1 OWNER

VIA set1 OWNER

{ SYSTEM
CURRENT
identifier1
VALUE OF identifiers2 IS identifiers3 }

[VIA set2 OWNER VALUE OF identifiers4 IS identifiers5]

where

record is a member record of set1;

set1 is the set being defined by the set description section;

identifiers2 is a list of data items in the owner record of set1;

identifiers3 is a list of data items whose contents are values contained in the data items of the corresponding elements of identifiers2;

set2 must have an owner record that is a member record of set1;

identifiers4 is a list of data items in the owner record of set2;

identifiers5 is a list of data items whose contents are values contained in the data items of the corresponding elements of identifiers4.

6. Record Section

The purpose of the record section is to define the records of the schema that are to be included in the subschema.

The format of the record section is divided into two entries.

Record Description

[Data Description]

The format for the Record Description is

01 record name

[WITHIN clause]

The Data Description has the following format

level number databasename

PICTURE clause

USAGE clause

OCCURS clause

RANGE clause

where all clauses are optional.

a. WITHIN clause

Function is to define and restrict selection of occurrences of the record named.

The format is

WITHIN realm

where realm is a realm name in the schema.

b. PICTURE clause

The format is identical to PICTURE clause of schema DDL record entry.

c. USAGE clause

Function is to specify format of COBOL data item.

The format is

```
USAGE IS { BIT          }
         { COMP-n       }
         { DISPLAY      }
         { DISPLAY-n    }
         { DATABASE-KEY }
         { INDEX        }
         { INDEX-n      }
```

d. OCCURS clause

The format is identical to the OCCURS clause of schema DDL record entry.

e. RANGE clause

The format is identical to RANGE clause of schema DDL record description.

V. DML

The Data Manipulation Language for DDBMS is specified in this chapter. The DML verbs which are used to extend the COBOL host language are presented. Appendix C gives the DML syntax.

A DML program is in fact an augmented COBOL program. Thus the overall structure is that of a COBOL program--Identification division, Data division, and Procedure division. Each division of the DML program includes standard COBOL features. These features are not discussed here. ONly the DML extensions are presented.

1. Data Division

The lone DML extension to the COBOL Data Division is the Schema Section. The Schema Section consists of an INVOKE clause which is used to call in the appropriate sub-schema for the application program.

a. INVOKE clause

Function is to identify sub-schema to be accessed by DML program.

The format is

INVOKE sub-schema OF SCHEMA schema

where sub-schema is a valid sub-schema name, and schema is the name of the schema containing sub-schema.

2. Procedure Division

The following verbs have been added to COBOL in the DML to provide for a full range database operations:

ACCEPT

CONNECT

DISCONNECT

ERASE

FIND

FINISH

GET

IF

MODIFY

OBTAIN

READY

STORE

USE

a. ACCEPT

The ACCEPT statement causes the contents of the specified currency

indicators to be accessed and provides a means of deriving the realm name of a data base key.

There are two formats:

i. ACCEPT identifier1 FROM [record / realm / set] CURRENCY

ii. ACCEPT identifier2 FROM [record / set / identifier3] REALM - NAME

where

identifier1 and identifier2 are data base keys;

record is a record name;

realm is a realm name;

set is a set name;

identifier2 is an alphanumeric data item.

b. CONNECT

The CONNECT statement causes a record stored in the data base to become a member of one or more sets in which the record has been defined as a possible member.

The format is:

CONNECT record TO {sets / ALL}

where

record is the record name;

sets is a list of sets of which record may be a member;

ALL means record will become a member of all possible sets.

c. DISCONNECT

The DISCONNECT statements removes a record from one or more sets.

The format is

DISCONNECT record FROM {sets / ALL}

where

record is the record name;

sets is a list of sets of which record is a member;

ALL means record is removed from all sets of which it is a member.

d. ERASE

The ERASE statement removes a record from the data base.

The format is

ERASE record [PERMANENT | SELECTIVE | ALL] MEMBERS

where

record is the record name;

PERMANENT means that all permanent members of the set owned by the current record of the run-unit will be erased. Transient members of the affected sets are treated as though they were DISCONNECTed;

SELECTIVE means that actions identical to that of the SELECTIVE option are taken except that when a transient member that is not a member of any other set is encountered, it is removed from the data base;

ALL means that all members of sets owned by the current record of run-unit are removed from the data base along with all members of any sets owned by records being removed from the data base by the ERASE command.

e. FIND

The function is to establish a record occurrence as current of run-unit, realm, record-name, and set. The option exists for retaining all currency indicators except current of run-unit.

The format of the FIND statement is

FINDrse [RETAINING CURRENCY FOR { MULTIPLE / RECORD / REALM / SET / sets }]

where rse is a record selection expression;

sets is a list of sets in the sub-schema;

MULTIPLE means no currency indicator except run-unit will be updated.

The record selection expression may have one of the following formats:

i. DBKEY IS identifier1

ii. { ANY / DUPLICATE } recordname1

iii. DUPLICATE WITHIN set1 USING { identifiers2 }

iv. { NEXT / PRIOR / FIRST / LAST / integer1 / identifier3 } [recordname2] WITHIN { set1 / realm1 }

v. CURRENT [recordname2] [WITHIN { set1 / realm1 }]

vi. OWNER WITHIN set2

vii. recordname2 WITHIN set1 [CURRENT]
[USING { identifiers2 }]

where

identifier1 must be a data base key;

recordname1 must have a location mode of CALC;

set1 is a set in the sub-schema;

identifiers2 is a list of identifiers defined in the current record of set1;

integer1 is an integer;

identifier3 is an integer data item;

recordname2 is the current record of set2 and realm1;

realm1 is a realm in the sub-schema;

set2 must not be a singular set;

ANY means that in case of several records with CALC keys equal to that of recordname1, the record with the lowest data base key and identical CALC key is selected;

DUPLICATE means that in case of several records with CALC keys equal to that of recordname1, the record with the data base key with the identical CALC key occurring next after the current of run unit is selected.

f. FINISH

The FINISH statement terminates the availability of one or more realms to the program.

The format is

FINISH realms.

where

realms is a list of realms in the sub-schema.

g. GET

The function is to transfer the contents of a record occurrence into the User Working Area.

The format is

GET [identifiers];

where

identifiers is a list of data base identifiers which are contained in the current record of run-unit.

h. MODIFY

The MODIFY statement is used to alter the contents of one or more data items in the current of run-unit record or to change the set

membership of the record.

There are two formats.

i. MODIFY

[{INCLUDING / ONLY} {ALL / sets} MEMBERSHIP]

ii. MODIFY identifiers

[INCLUDING {ALL / sets} MEMBERSHIP]

where

sets is a list of sets of which the current of run-unit record is a member;

identifiers is a list of data items contained in the current of run-unit record;

ONLY means that the contents of data items are not changed.

i. OBTAIN

The OBTAIN statement performs the function of a FIND statement immediately followed by a GET statement.

The formats for OBTAIN are identical to those of FIND.

j. READY

The READY statement prepares a realm for processing.

The format is

READY realm

[USAGE MODE IS [EXCLUSIVE / PROTECTED] {RETRIEVAL / UPDATE}]

where

realm is the name of a realm in the sub-schema.

k. STORE

The STORE statement causes a record to be stored in the data base.

The format is

STORE record

[RETAINING CURRENCY FOR { MULTIPLE / REALM / SET / sets / RECORD }]

where

record is a record in the sub-schema;

sets is a list of sets of which record is a member;

MULTIPLE means that all currency indicators except run-unit will be retained.

1. USE

The USE statement specifies procedures that produce privacy keys.

The format is

USE FOR PRIVACY

[[ON { EXCLUSIVE / PROTECTED } { RETRIEVAL / UPDATE } FOR { realms / REALMS }]
[ON { CONNECT / DISCONNECT / STORE / ERASE / ERASE PERMANENT / ERASE SELECTIVE / ERASE ALL / GET / MODIFY / FIND / OBTAIN } FOR { records / RECORDS }]
[,ON { CONNECT / FIND / OBTAIN / DISCONNECT } FOR { sets / SETS }]
[,ON { GET / MODIFY / OBTAIN / STORE } FOR identifiers]]

where

realms is a list of realms in the sub-schema;

records is a list of records in the sub-schema;

sets is a list of sets in the sub-schema;

identifiers is a list of identifiers in the sub-schema.

REALMS, SETS, or RECORDS implies a privacy key will be generated for all realms, sets, or records in the sub-schema.

DMCL

A. Overview

The CODASYL committee has not specified a standard Device Media Control Language. The function of the DMCL is to map the logical data base to a physical storage facility. This type of mapping is typically carried out by job control language statements in an application program. In the DDBMS, the purpose of the DMCL is to spare the applications programmer and data base administrator the trauma of understanding the job control languages of all machines that may be involved in accessing data. Each application program has a set of DMCL instructions associated with it by the DBA. The DBA produces high level DMCL statements which are mapped into job control statements for the appropriate machines by the DMCL translator.

The DMCL for this system is intended to be easy to use and understand. It differs from existing DMCL's [6] in that the machine controlling access to the data must also be specified. The remainder of this section defines the function and format of the DDBMS DMCL. Appendix D contains the DMCL syntax.

B. Specifications

A DMCL program is composed of four sections as shown below.

DMCL Description

BUFFER Description

REALM Description

PROCESSOR Description

1. DMCL Description

The purpose of the DMCL description is to name the DMCL segment and identify the schema which is being mapped to physical storage.

The format is

DD dmclname FOR schemaname

where

dmclname is the name of DMCL segment;

schemaname is the schema being described.

2. Buffer Description

The buffer description is intended to assign a name and size to the system buffer area and denote the buffer page size.

The format is

BD name

PAGE SIZE IS integer1 BYTES

BUFFER SIZE IS integer2 PAGES

where

name is the buffer name;

integer1 and integer2 are integers.

3. Realm Description

The realm description lists the realms in the schema that are mapped in this DMCL segment.

RD {realms / ALL}

where

realms is a list of realms in the schema;

ALL means all realms of the schema are to be included in the DMCL segment.

4. Processor Description

The processor description section lists the machines which have access

to the schema.

The format is

BACKEND processorid

HOSTS {processorids / ALL}

where

processorid is the network identifier of the processor controlling access to the schema;

processorids is a list of the network identifiers of processors upon which application programs that reference the schema can reside;

ALL means that any processor in the network (including future configurations) may reference the schema.

VII. Utilities

A. Overview

All CODASYL based data base systems consist of the same basic functional components, schema DDL, sub-schema DDL, DML, DMCL. (Although in some cases the functions of two languages are combined into a single language [7]). One important area in which this uniformity evaporates is data base utilities. The CODASYL committee suggested a rather brief list of utility functions but did not formally incorporate any into their specifications.

The utilities of a DBMS have a major impact on the useability and overall performance of the system. A powerful set of utility programs permits the DBA to carry out his/her duties in a precise and efficient manner. Since the DDBMS is physically a more complex system than any other CODASYL-derived DBMS, an extensive set of utilities programs is included in the system. Many of the utilities are for the use of the DBA. However several are intended to facilitate the production of application programs. Certain utilities are available at execution time (such as those involved in recovery) while others are off line functions (i.e. changing page sizes). The utilities will

be listed below, grouped by type of usage.

B. Off Line Functions

1. Dump to tape - This routine is used to transfer a sub-schema realm, set, or record to a tape file. Each dump label indicates the name and type of data being transferred. The date of the transfer is included in the header for archival purposes.

2. Print - This routine allows the printing of any portion of the data base down to the record level. The format of the records is as specified in the schema. Sets are printed in their defined order.

3. Edit - The editing utility is employed to modify the values of data items in the data base. The structure of the data base is not changed. That is, sets or records may not be added or deleted. However, set relationships may be altered as a result of changes to data items.

4. Initialization - This routine is used to establish a data base on secondary storage. The format of the data base is defined by the object sub-schema produced by a sub-schema DDL compile. The data is loaded from either cards or tape using a specified format.

5. Data Base Reorganization - This routine modifies the physical structure of the data base. Among the operations performed by this utility are changing page size, altering the placement of overflow or index pages, moving realms or sub-schemata to different physical devices, and transferring control of portions of the data base (not lower than the realm level) to another processor in the network. The primary motive for using the reorganization routine is to attempt to fine tune the performance of the DDBMS by better distributing the data to avoid secondary storage access bottlenecks. The information upon which the reorganization is based is derived from the statistical analysis routine (see VII B.7).

6. Garbage collection - When a substantial number of additions and

deletions have been performed on the data base, secondary storage may become highly segmented with many small pockets of unused space. In order to improve the efficiency of secondary storage utilization, a garbage collection routine can be employed to compact the used and free space on each page. The result of the garbage collection is to create more locations for records on the data pages. Each garbage collection is targeted for a particular secondary storage device. Information concerning the need for garbage collection is obtained from the statistical analysis utility described in the next paragraph.

7. Statistical analysis - The DBA's guide to the performance of the DDBMS is the statistical analysis utility. This routine tabulates and summarizes the statistical information gathered by the on-line data base activity logging routine (Section VII C.1). Among the performance descriptors processed by this utility are the total and average number of DML operations, reads, writes, and currency of set references per schema, subschema, realm, set and record; page content and distribution statistics; average and total numbers of page accesses and overflows; number of collisions per CALC value; and queuing information for DBMS host and back-end tasks, communication facilities, and secondary storage devices.

8. Integrity check - In a DBMS, perhaps the most difficult errors to detect are those which produce scattered erroneous results but which do not cause catastrophic failures. Because of the large amount of software and hardware resources necessary to realize the DDBMS, there is a definite likelihood that minor errors may periodically occur. The integrity check utility is executed in order to verify the absence of a certain class of errors in the data base or to locate portions of the data base that contain improper information. Two types of information, data and links, are verified in the integrity check. Data items that

have the RANGE attribute are tested to insure compliance with the established limits. Set linkages are verified along with CALC chain linkages which have been built up due to collisions on CALC keys.

9. Restore from Tape - This routine allows the reconstruction of all or part of a data base from a tape created by the Dump to Tape utility (VII B.1).

10. Security Violation - The security violation utility is an on-line routine under the control of the DBA. It is called by the security monitor (VII C. 2) whenever a security violation is detected. The security violation utility determines the action to be taken in response to the unauthorized attempt at data base usage. Among the possible responses are recording user id and illegal action, displaying a message to the user, terminating and rolling back the offending process, or notifying the operator of the point of origin of the illicit task so that the user may be contacted by security personnel.

11. Manipulate Locks - A special version of the editing utility is used to alter privacy locks. This routine can only be used by the DBA .

C. On-Line System Functions

1. Data Base Activities Logging - The log routine is a multifaceted recorder of data base activity. Information such as DML commands, page images, and statistical data is accumulated for rollback, recovery, and analysis purposes. The information is recorded onto either a tape or disk file whenever a DML operation is processed by either a host or back-end machine. The reason for incorporating both statistical and operational information into a single file is that the requirement for separate physical devices may be prohibitive in a minicomputer configuration.

2. Security Monitor - The security monitor is activated when a user-

supplied privacy key fails to match the DBA-specified lock. As indicated in the description of the schema DDL, privacy locks can be established at all levels in the DDBMS. Upon detection of an illicit access attempt, the security monitor routine calls the security violation utility (VII B.10) which handles the intrusion in an appropriate manner.

3. Realm Enable/Disable - The DBA may selectively disable all access to a realm and later re-enable access by means of these routines.

4. Rollback - The rollback utility is employed to reverse the effect of a single DML command, a sequence of commands, or a complete task. During rollback, the affected realms are first disabled by the realm disable utility (VII C.3). The effects of all DML commands specified for rollback are then reversed. Realms are enabled when they are not affected by any command waiting to be retracted.

5. Checkpoint - The optional checkpoint utility involves the periodic dumping of a sub-schema to back-up storage. The checkpoint utility is activated whenever a sub-schema is invoked by an application task and upon termination of that task. During a checkpoint dump, all realms of the sub-schema must be disabled. The checkpoint dump also contains status information and the UWA for all active user tasks.

6. Recovery - The recovery routine is initiated when the DDBMS is determined to have operated erroneously. If the checkpoint option (VII C.5) has been employed, the affected realms are restored from their most recent checkpoints. All active tasks accessing those realms are restored to the status they held at the time of the checkpoints.

7. Restart - In the case of catastrophic failure, the data base must be completely restored from a Dump Tape (VII B.9) and all user tasks must be restarted. New checkpoint and audit files must be created.

8. Encryption - For purposes of security, the stored data may be encoded so that illicit access without the proper decoding operations

would be without benefit to the interloper. Use of the decoding operation must be carefully restricted by privacy locks.

9. Data compression - Since storage is a critical resource in the DDBMS, a data compression utility has been included to reduce the secondary storage requirements of data bases. The data compression and expansion introduce some time overhead while producing a decrease in secondary storage requirements.

10. Trace - The trace utility serves as a debugging aid to application programmers. It can be activated and deactivated under program control. The trace output indicates the statements executed, data transferred, status conditions, set contents, values of CALC keys, currency indicators, etc.

VIII. Special Features

A. Introduction

Many features may be added to a basic CODASYL DBMS to enhance its useability. In this chapter, six such additions, Query Language, Report Writer, Data Dictionary, Data Base Administrator Language, Debugging Facility, and an Interactive Data Base Design Language are discussed. The initial version of DDBMS will not include any of these enhancements. They will be incorporated into the DDBMS as soon as possible.

B. Query Language

A query language provides the unsophisticated user with the ability to access the data base interactively in an easy to use format. The query language permits information in any sub-schema to be operated upon by commands input at a terminal. Naturally all privacy requirements must be met before such access can transpire. The organization of the software comprising the query language facility is shown in Figure 6. In essence, each query language command is translated into one or more DML commands which access the data base

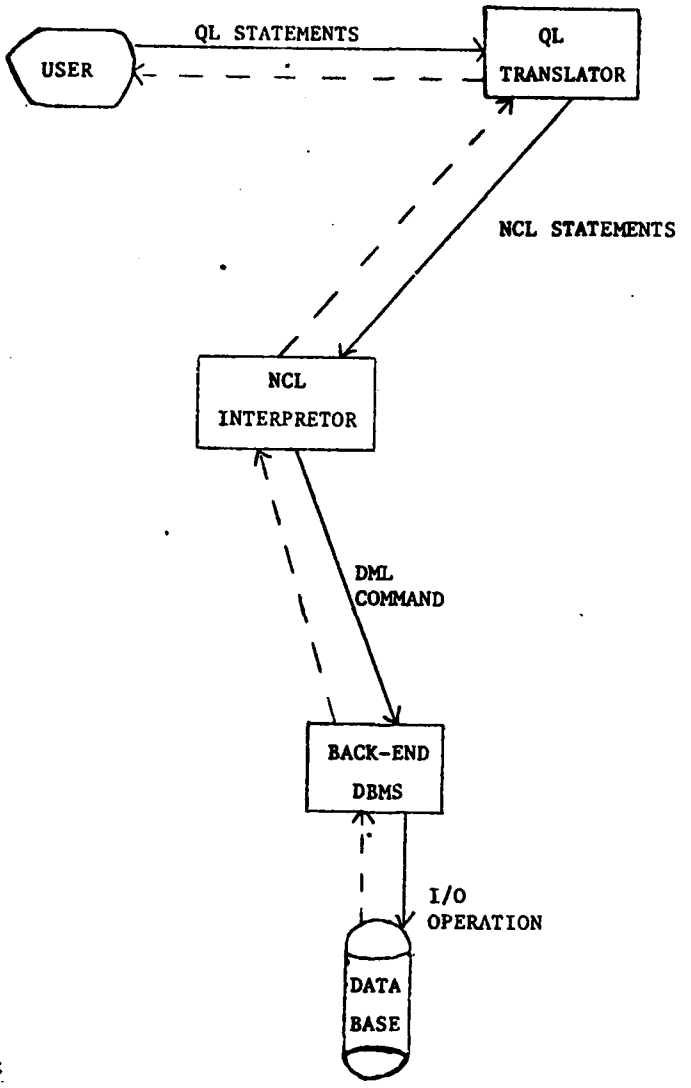


Figure 6

Query Language Software Structure

and a series of Network Control Language (NCL) [8] statements which initiate the data base access and control the transfer of information. Building the query language on the NCL and the DML facilitates the implementation and produces a generalized and portable facility.

C. Report Writer

In a data base system capable of maintaining a large variety of data, there is a requirement for many diverse reports. Rather than force the applications program to completely specify the format of each document to be generated, a report writer feature is to be incorporated in DDBMS. The report writer is sufficiently general to allow any information on the data base to be retrieved and printed in any format desired.

D. Data Dictionary

The data dictionary facility provides a complete description of the data base. It is used in the initialization and loading of the data base. The data dictionary contains the schemata, sub-schemata, and their associated DMCL segments which are extracted prior to each application program execution. A listing of the data dictionary will indicate the current structure of the data base.

E. Data Base Administrator Language

The DBA language is an extension of the query facility. The DBA language makes it possible for the DBA functional utilities (Sec. VII B.) to be initiated directly from a terminal by means of high-level commands. It is imperative that access to the DBA language be carefully limited since misuse of its powerful features could play havoc with the integrity of the data base.

F. Interactive Data Base Design Language

The initial description of the data base in terms of a schema involves considerable effort on the part of the DBA. A commonly used tool is the Bachman diagram (see Figure 7 for an example). Once a Bachman diagram has been developed

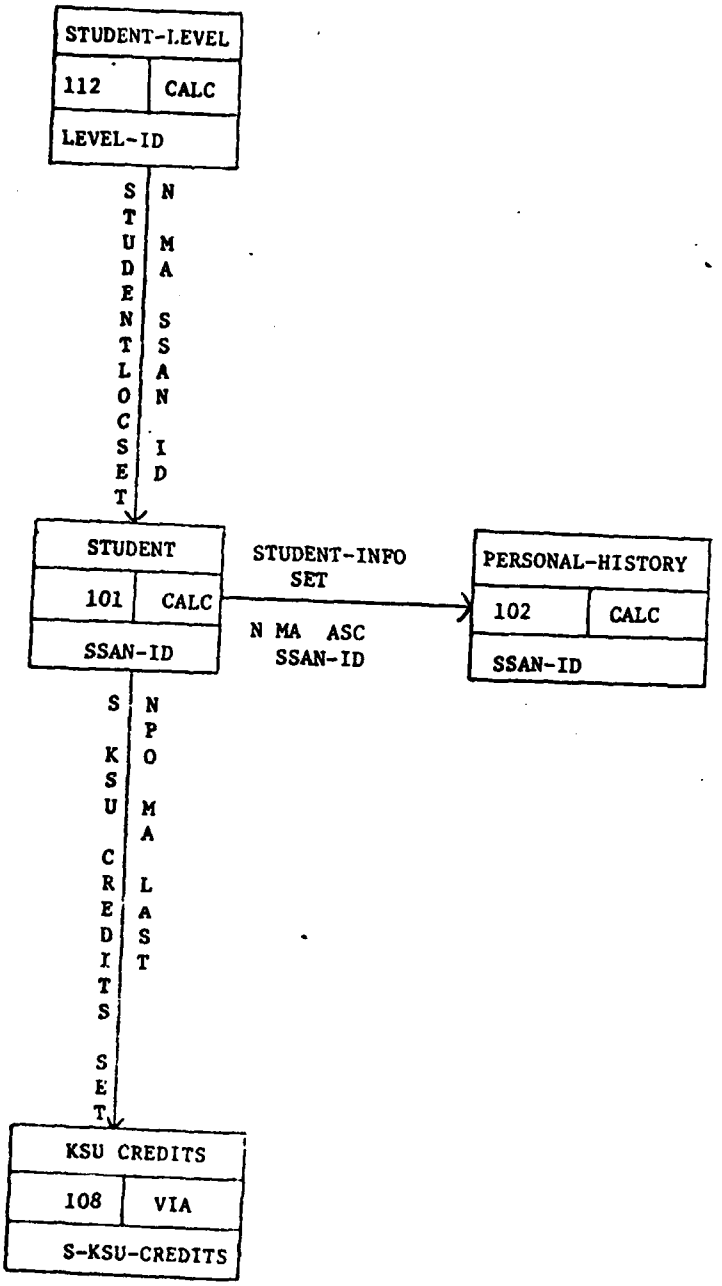


Figure 7

Sample Bachman Diagram

it is a relatively straightforward process to map to schema DDL statements. The Interactive Data Base Design Language (IDBDL) will allow the DBA to graphically input a Bachman diagram and have schema DDL code produced as output. Facilities for creating, deleting, and modifying diagrams are also included in IDBDL.

D. Debugging Facility

The debugging facility provides the capability to test a program without affecting a real data base. The debugger operates in an interactive environment to provide a maximum capacity for program testing. Among the features of the debugger are tracing, insertion and deletion of sets, records, and data items, breakpoints, and statistical output.

IX. DDBMS and Relational Data Bases

Currently, the two principal types of organization for commercially available data bases are network (CODASYL) or hierarchical (such as IBM's IMS). There are many systems which are hybrids of these two primary types. Hybrid systems include TOTAL, ADABAS, System 2000, INQUIRE. In virtually all commercially vended data base management systems, the user is aware of the physical structure of the data base.

The concept of a relational data base was initially proposed by E. F. Codd [9], also see References [10-12]. The method of physical organization of the data base is transparent to the user of a relational system. A relational DBMS is very well suited for the formulation of complex queries and the specification of intrinsic relationships among the data items. Many experts feel that the relational approach holds the key to future data base development [13]. The main drawback thus far has been the excessive amount of space required to maintain the tables specifying the relations.

If the potential of the relational method is realized, DDBMS and other CODASYL systems must have a means of bridging the gap between the relational

and network organizations. Some investigation has been conducted to determine which features of the CODASYL systems would be most compatible with the relational model [14]. The following features in the DDBMS can be used for the purpose of forming a bridge with the relational model.

1. OWNER IS SYSTEM.

2. ORDER IS IMMATERIAL.

3. LOCATION MODE IS SYSTEM.

4. SET MEMBERSHIP IS MANDATORY AUTOMATIC.

By making use of the features listed above and with an improved query language, DDBMS should be capable of functioning in a manner similar to a relational data base system.

X. Comparison With Other CODASYL DBMS

A. Overview

DDBMS is a larger subset of the CODASYL specifications than any commercially available DBMS. A detailed comparison of DDBMS with four CODASYL derivatives, DBMS-10, DMS-1100, EDMS, and IDMS is given in the Tables of Appendix E. The main differences between DDBMS and the other systems are expounded upon in the succeeding paragraphs.

B. Unique Features in DDBMS (among CODASYL DBMS systems)

1. LOCATION MODE IS SYSTEM - this feature is included for future compatibility with relational DBMSs.

2. Access locks for query store and modify - provided as part of extensive security system.

3. ORDER IS IMMATERIAL - a feature included for relational DBMS compatibility.

4. Security facilities at all levels of the data base.

C. Features Not Contained in CODASYL But Found in Other Network DBMSs

1. NEXT as default set linkage - saves coding in DDLs.

2. Storage specifications contained in schema DDL - allows a more

comprehensive definition of data base in the schema DDL.

3. INVOKE SUB-SCHEMA DML Verb - explicit request of sub-schema for application program.

4. OBTAIN DML Verb - combination of FIND and GET to facilitate retrieval.

D. CODASYL Features Omitted from DDBMS

1. Unlimited name length - restricted to 30 character maximum.

2. Sub-schema DDL may change privacy locks, reorder items - omitted to improve compatibility of sub-schema with schema.

3. KEEP/FREE DML Verbs - omitted to allow the system to control record access in order to avoid deadlock situations.

E. Utilities

Since the CODASYL specifications do not include utilities, absolute comparisons among the network systems are difficult. However, DDBMS has a more comprehensive utility package than any of the systems listed in Reference [7].

Bibliography

1. CODASYL COBOL 1973 Journal of Development, Dept. of Supply and Services, Technical Service Branch, Ottawa, Ontario (revisions to June, 1975).

2. Maryanski, F.J., et al., "A Minicomputer Based Distributed Data Base System," NBS-IEEE Trends and Applications Symposium: Micro/Mini Systems; May, 1976.

3. Wallentine, V.E. and Maryanski, F.J., "Implementation of a Distributed Data Base System," TR-CS-11-75, Computer Science Dept., Kansas State University, Manhattan, KS 66506, Nov., 1975.

4. Maryanski, F.J., "Functional Specifications of a Distributed Data Base Management System," Technical Report, Computer Science Dept., Kansas State State University, Manhattan, KS 66506, (in prep.).

5. Maryanski, F.J. and Wallentine, V.E., "Resource Sharing in a Distributed Data Base Management System," Technical Report, Computer Science Dept. Kansas State University, Manhattan, KS 66506, (in prep.).

6. IDMS Data Definition Languages, Utilities and GCI Reference Guide, Release 3.1 Revision 1, Cullinane Corp., Wellesley, Mass., April, 1975.

7. Warren, T., "Feature Analysis of CODASYL Data Base Management Systems," AD-A014-972, NTIS, Dept. of Commerce, Springfield, VA, June, 1975.

8. Wallentine, V.E., "Functionally Distributed Computing Systems," Technical Report, Computer Science Dept., Kansas State University, Manhattan, KS 66506, (in prep.).

9. Codd, E.F., "A Relational Model of Data for Large Shared Data Banks," CACM (13,6), June, 1970, pp. 377-387.

10. Codd, E.F., A Data Base Sublanguage Founded on the Relational Calculus, Proc. 1971 ACM SIGFIDET Workshop, Nov., 1971, pp. 35-68.

11. Date, C.J., Introduction to Data Base Management, Addison Wesley, Reading, Mass., 1975.

12. Boyce, R.F., et al., "Specifying Queries as Relational Expressions: The SQUARE Data Sublanguage," CACM (18,11), Nov., 1975, pp. 621-628.

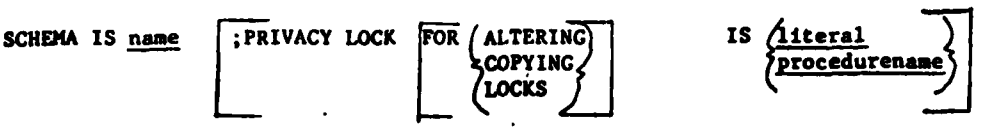
13. Klimbie, J.W. and Koffeman, K.L. (eds.), Data Base Management, North Holland, April, 1974, (see post-paper discussions).

14. Nijssen, G.W., "Data Structuring in the DDL and Relational Model," in Data Base Management, Klimbie, J.W. and Koffeman, K.L. (eds.), North Holland, April, 1974, pp. 363-384.

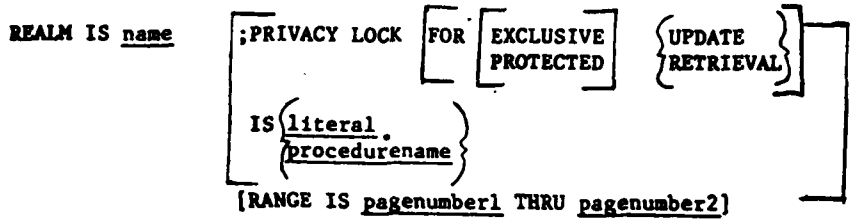
Appendix A

Schema DDL

Schema Section



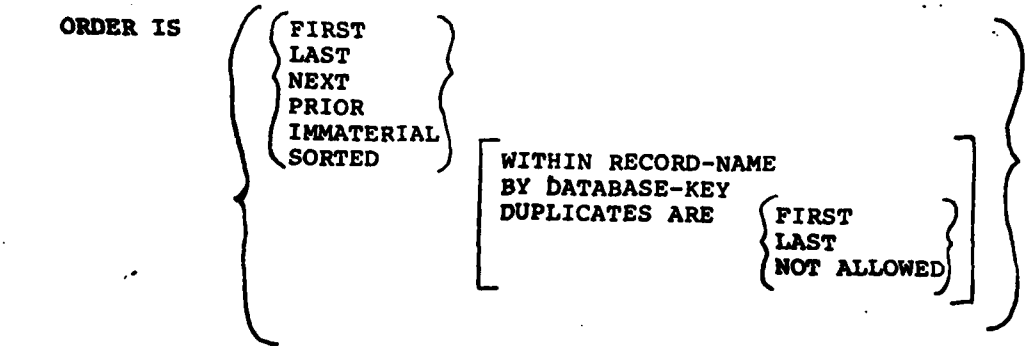
Realm Section



Set Section

Set Description

Set IS name



```
[;PRIVACY LOCK [FOR {OBTAIN   }]  IS {literal     }]
[                  {FIND      }]      {procedurename}
[                  {CONNECT   }]
[                  {DISCONNECT}]
```

```
OWNER IS record name
Member section
MEMBER IS name  {MANDATORY} {AUTOMATIC}
                {OPTIONAL } {MANUAL   }

  [LINKED TO OWNER] [DUPLICATES ARE NOT ALLOWED
                        FOR identifiers          ]

  {ASCENDING } KEY IS identifiers [DUPLICATES ARE {FIRST      }]
  {DESCENDING}                                    {LAST       }
                                                  {NOT ALLOWED}]

   SET SELECTION IS THRU {CURRENT OF SET                }
                         {LOCATION MODE OF OWNER        }
                         {  [USING identifiers]         }
```

```
Record Section
   Record Description
      RECORD IS name

         {LOCATION MODE IS DIRECT dbkey                }
         {LOCATION MODE IS CALC USING keys             }
         {    DUPLICATES ARE [NOT] ALLOWED             }
         {LOCATION MODE IS VIA setname SET             }
         {LOCATION MODE IS SYSTEM                      }

          WITHIN realm

          [PRIVACY LOCK [FOR operations]  IS {literal     }]
          [                                  {procedurename}]
```

Data Description

[level number]

```
{ PICTURE IS                character-string      [DEPENDING ON identifier]
{ TYPE IS      [BINARY ] [FIXED ]  [REAL   ]      precision
               [DECIMAL] [FLOAT ]  [COMPLEX]
{ TYPE IS      {BIT      }  size
               {CHARACTER}
{ TYPE IS      DATABASE - KEY
```

[OCCURS count time]

[RANGE IS FROM value1 TO value2]

```
[PRIVACY LOCK IS  [FOR {STORE   } IS {literal      }]
                       {GET     }     {procedurename}
                       {MODIFY  }
                       {OBTAIN  }]
```

Appendix B

Sub-Schema DDL Syntax

Sub-Schema Section

```
    SS    sub-schema name  WITHIN  schema-name
[;PRIVACY LOCK   [FOR  {LOCKS    }]  IS  {  literal     }]
                       {ALTERING }       {  procedurename}
                       {INVOKING }
.
[;PRIVACY KEY IS  {literal     }]
                  {procedurename}
```

Alias Section

```
ALIAS OF name   [REALM-NAME]   IS   name2
                [SET-NAME  ]
                [RECORD-NAME]
```

Realm Section

```
   RD  {ALL       }
       {realm-name}
```

Set Section

```
   SD  {ALL    }
       {setname1}
```

```
;SET SELECTION FOR  recordname IS
 VIA setname1 OWNER  {SYSTEM                                    }
                     {CURRENT                                   }
                     {identifier1                               }
                     {VALUE OF identifiers2 IS identifier3      }
 [VIA setname2 OWNER VALUE OF identifier4  IS identifier5]
```

Record Section

```
   01   recordname  [WITHIN realm]
```

Data Description

level number dataname1

[; PIC IS characterstring]

[; [USAGE IS] { BIT | COMP | COMP-n | DBKEY | DISPLAY | DISPLAY-n | INDEX | INDEX-n }]

[; OCCURS integer TIMES]

[{ ASCENDING | DESCENDING } KEY IS {datanames2}
  [INDEXED BY indices]]

APPENDIX C

DML Syntax

```
Data Division
     INVOKE sub-schema OF SCHEMA schema

Procedure Division
     ACCEPT identifier1 FROM { [record]  CURRENCY
                             [realm]
                             [set]
                             [record]    REAL-NAME
                             [set]
                             [identifier2] }

CONNECT [record] TO {sets}
                    {ALL}

DISCONNECT [record] FROM {sets}
                         {ALL}

ERASE [record] [PERMANENT]  MEMBERS
               [SELECTIVE]
               [ALL]

FIND record selection expression

[; RETAINING CURRENCY FOR {MULTIPLE}
                          {REALM   }
                          {SET     }
                          {sets    }
                          {RECORD  }]
```

```
FINISH realms

GET [identifiers]

MODIFY { [record]    [ {INCLUDING} {ALL}   MEMBERSHIP ] }
       {             [ {ONLY     } {sets}  ]           }
       { identifiers [ INCLUDING   {ALL}   MEMBERSHIP ] }
       {             [             {sets}  ]           }

OBTAIN record selection expression

[; RETAINING CURRENCY FOR {MULTIPLE}]
                          {REALM   }
                          {SET     }
                          {sets    }
                          {RECORD  }

READY [ realm [ USAGE-MODE IS [EXCLUSIVE] {RETRIEVAL}] ]
              [               [PROTECTED] {UPDATE   }  ]

STORE record

[; RETAINING CURRENCY FOR {MULTIPLE}]
                          {REALM   }
                          {SET     }
                          {sets    }
                          {RECORD  }
```

```
USE FOR PRIVACY  [ON  [EXCLUSIVE]  {RETRIEVAL}  FOR  {realms}]
                      [PROTECTED]  {UPDATE   }       {REALMS}

[,ON  {CONNECT         }     FOR  {records}]
      {DISCONNECT      }          {RECORDS}
      {STORE           }
      {ERASE           }
      {ERASE PERMANENT }
      {ERASE SELECTIVE }
      {ERASE ALL       }
      {GET             }
      {MODIFY          }
      {FIND            }
      {OBTAIN          }

[,ON  {CONNECT   }           FOR  {sets}]
      {FIND      }                {SETS}
      {DISCONNECT}
      {OBTAIN    }

[,ON  {GET   }               FOR  identifiers]
      {MODIFY}
      {STORE }
      {OBTAIN}
```

There are seven formats for record selection expression.

```
1.  DBKEY  IS  identifier1

2.  {ANY      }  recordname1
    {DUPLICATE}

3.  DUPLICATE WITHIN set1 USING  identifiers2

4.  {NEXT       }  [recordname2]  WITHIN  {set2  }
    {PRIOR      }                         {realm1}
    {FIRST      }
    {LAST       }
    {integer    }
    {identifier3}

5.  CURRENT [recordname2] [WITHIN  {set2  }]
                                   {realm1}

6.  OWNER  WITHIN  set2

7.  recordname2 WITHIN set2  [CURRENT] [USING identifiers2]
```

APPENDIX D

DMCL Syntax

DMCL Description

DD dmclname FOR schema

Buffer Description

BD buffer

PAGE SIZE integer1 BYTES
BUFFER SIZE integer2 PAGES

Realm Description

RD { realms / ALL }

Processor Description

BACK_END processor id
HOSTS { processor ids / ALL }

APPENDIX E

Comparison of DDBMS and Other CODASYL Systems

R

The comparison between DDBMS and the other CODASYL data base systems is made here using tables that indicate the main features of several systems. The systems considered are DDBMS, CODASYL specifications, DEC's DBMS-10, UNIVAC's DMS-1100, Honeywell's (Xerox) EDMS, and Cullinane's IDMS. The tables are an extension of those prepared by Warren [7].

ATTRIBUTES	DBA	CODASYL	DBMS-10	DMS-1100	EDMS	IDMS
1. Data Structures						
1.1 Realms	Realm	Realm	Area	Area	Area	Area
Temporary realms	NO	YES	YES	NO	NO	NO
Name	1-30 char.		1-30 char.	1-12 char.	1-30 char.	1-16 char.
Security	Privacy Locks	Privacy Locks	Privacy Locks	O/S supplied	O/S supplied	Privacy Locks
Usage Modes	retrieval update protected r/u exclusive r/u	retrieval update protected r/u exclusive r/u	retrieval protected r/u exclusive u	retrieval update protected r/u exclusive r/u initial load	retrieval update shared ret shared upd. create	retrieval update exclusive r/u protected r/u
1.2 Records	Record	Record	Record	Record	Record	Record
Name	1-30 char.	1-30 char.	1-30 char.	1-30 char.	1-16 char.	
Uniqueness	Among records of schema	Among records	Among records of schema	Among records	Among group, item sets	Within data base
Internal id	DB-KEY	DB-KEY	DB-KEY	DB-KEY NUMERIC ID	DB-KEY INTEGER-ID	DB-KEY NUMERIC-ID
Security	access locks	access locks	none	none	privacy locks	privacy locks
Location Modes	DIRECT CALC VIA SYSTEM	DIRECT CALC VIA SYSTEM	DIRECT CALC VIA	DIRECT CALC VIA INDEX SEQ.	DIRECT CALC VIA INDEXED	DIRECT CALC VIA
1.3 Items	Data Item	Data Item	Data Item	Data Item	Data Item	Data Item
Name	1-30 char.		1-30 char.	1-30 char.	1-30 char.	1-16 char.
Uniqueness	within record	within record	within record	within record	within group among set, group names	within record
Data Aggregates						
Vector	YES	YES	YES	YES	YES	YES
Repeating groups	YES	YES	YES	YES	YES	YES

ATTRIBUTES	DDBMS	CODASYL	DBMS-10	DMS-1100	EDMS	IDMS
TYPES	arithmetic character DB-KEY	arithmetic character DB-KEY implementer defined	arithmetic character DB-KEY	arithmetic lock display DB-KEY area-key area-range	arithmetic character	arithmetic character
Security	access locks query,store, modify	access locks query,store, modify	none	none	privacy locks retrieval, update	none
1.4 Sets	set	set	set	set	set	set
Names	1-30 char.		1-30 char.	1-30 char.	1-30 char.	1-16 char.
Uniqueness	among set names of schema	among set names of schema	among set names	among set names	among set, group names	to data base
Mode						
chain	YES	YES	YES	YES	YES	YES
Pointer array	YES	YES DYNAMIC	NO	YES	NO	NO
LINKAGE						
NEXT	YES(DEFAULT)	YES	YES(DEFAULT)	YES(DEFAULT)	YES(DEFAULT)	YES(DEFAULT)
PRIOR	YES	YES	YES	YES	YES	YES
OWNER	YES	YES	YES	YES	YES	YES
Order of Insertion						
FIRST	YES	YES	YES	YES	YES	YES
LAST	YES	YES	YES	YES	YES	YES
NEXT	YES	YES	YES	YES	YES	YES
PRIOR	YES	YES	YES	YES	YES	YES
IMMATERIAL	YES	YES	NO	NO	NO	NO
Sort Keys	DB-KEY record name data items w/i record type	DB-KEY record name data items w/i record type	DB-KEY record name data items w/i record type	DB-KEY data items, record name	data items group number	data items
Ascending/Descending	a/d	a/d	a/d	a/d	a/d	a/d

ATTRIBUTES	[illegible]	CODASYL	[illegible]	[illegible]	EDMS	IDMS
Duplicates						
FIRST	YES	YES	YES	YES	YES	YES
LAST	YES	YES	YES	YES	YES	YES
NOT ALLOWED	YES	YES	YES	YES	YES	YES
Membership						
Manual	YES	YES	YES	YES	YES	YES
Automatic	YES	YES	YES	YES	YES	YES
Optional	YES	YES	YES	YES	YES	YES
Mandatory	YES	YES	YES	YES	YES	YES
SET OCCURENCE SELECTION						
Current of Set	YES	YES	YES	YES	YES	NO
Location mode of Owner	YES	YES	YES	YES	YES	NO
Aliases	YES	YES	YES	YES	YES	NO
2. DDL						
Schema DDL	YES	YES	YES	YES	YES	YES
Contain Storage Structure specs	YES	NO	YES	YES	YES	YES
Separate DMCL	YES	YES	YES	NO	NO	YES
Sub-schema DDL	YES	YES	YES	NO	YES	YES
Subsets data base	YES	YES	YES	NO	YES	YES
Renames elements	YES	YES	NO	NO	YES	NO
Privacy lock change	NO	YES	NO	NO	NO	YES
Reorder items	NO	YES	NO	NO	NO	NO
3. DML						
3.1 Host Language	COBOL	COBOL	COBOL FORTRAN	COBOL	COBOL, FORTRAN META-SYMBOL	COBOL, PL/I FORTRAN, ASSEMBLER
Method of Invocation	verb	verb	verb	verb	call	verb call

ATTRIBUTES	DDBMS	CODASYL	DBMS-10	DMS-1100	EDMS	IDMS
3.2 Control Statements						
Invoke subschema	INVOKE		INVOKE	INVOKE		INVOKE
Open realm	READY	READY	OPEN	OPEN	open routines	OPEN
Close realm	FINISH	FINISH	CLOSE	CLOSE	close routines	CLOSE
3.3 Retrieval Statements						
Locate	FIND	FIND	FIND	FIND	find routines	FIND
Locate & access	OBTAIN			FETCH		OBTAIN
Access	GET	GET	GET	GET	GET	GET
Lock/Release		KEEP/FREE		KEEP/FREE		
Currency Save	ACCEPT	ACCEPT	MOVE	MOVE		MOVE
3.4 Modification Statements						
Add	STORE	STORE	STORE	STORE	STORE	STORE
Change	MODIFY	MODIFY	MODIFY	MODIFY	MODIFY	MODIFY
Delete	ERASE	ERASE	DELETE	DELETE	delete routines	DELETE
Reorder Sets		ORDER				
Reorganize Sets	CONNECT DISCONNECT	CONNECT DISCONNECT	INSERT REMOVE	INSERT REMOVE	Link,Delink, Relink	INSERT REMOVE
3.5 Special Statements						
Error Control	USE	USE			error routines	
Statistics	stat. utilities				stat. routines	
Trace	trace utility			LOG	trace routines	
4. Security						
Method	privacy lock/key	privacy lock/key	lock/key	none	password lock/key	privacy locks in sub-schema
Level	all	all	schema, sub-schema area	none	group, item level	area,set record level

ATTRIBUTES	DDBMS	CODASYL	DBMS-10	DMS-1100	EDMS	IDMS
Category	query,store, modify	query,store, modify.	open	none	retrieve, update	DML verbs